12. Machine Learning Methodology in Bioinformatics

Colin Campbell

Machine learning plays a central role in the interpretation of many datasets generated within the biomedical sciences. In this chapter we focus on two core topics within machine learning, *supervised* and *unsupervised* learning, and illustrate their application to interpreting these datasets. For supervised learning, we focus on *support vector machines* (SVMs), which is a subtopic of *kernel-based learning*. Kernels can be used to encode many different types of data, from continuous and discrete data through to graph and sequence data. Given the different types of data encountered within bioinformatics, they are therefore a method of choice within this context. With *unsupervised learning* we are interested in the discovery of structure within data. We start by considering hierarchical cluster analysis (HCA), given its common usage in this context. We then point out the advantages of *Bayesian* approaches to unsupervised learning, such as a principled approach to model selection (how many clusters are present in the data) through to confidence measures for assignment of datapoints to clusters. We outline five case studies illustrating these methods. For supervised learning we consider prediction of disease progression in cancer and protein fold prediction. For unsupervised learning we apply HCA to a small colon cancer dataset and then illustrate the use of Bayesian unsupervised learning applied to breast and lung cancer datasets. Finally we consider *network inference*, which can be approached as an unsupervised or supervised learning task depending on the data available.

In this chapter we consider the application of modern methods from machine learning to the analysis of biomedical datasets. There are a substantial number of machine learning methods which could be used in the context of bioinformatics, and so we are necessarily selective. We focus on the two commonest themes within machine learning, namely *supervised* and *unsupervised* learning. Many methods have been proposed for supervised learning, and so, in Sect. 12.1, we choose to concentrate on *kernel-based methods*, specifically *support vector machines* (SVMs). SVMs are a popular approach to classification and have several advantages in handling datasets from bioinformatics. In particular, biomedical data can appear in many forms from continuous-valued and discrete data to network structures and sequence data. These different types of data

can be encoded into *kernels* which quantify the similarity of data objects. We start by introducing classification and the support vector machine for binary classification. In Sect. 12.1.1 we then extend this approach to multiclass classification, learning in the presence of noise, the association of confidence measures to class labels, and regression (i. e., using continuously valued labels). In Sect. 12.1.3 we consider simple kernels, complex kernels for graphs, strings, and sequences, and *multiple kernel learning*, where we build a decision function for prediction using multiple types of input data.

The second area of machine learning we consider is *unsupervised learning*, where we are interested in the discovery of structure in data. In Sect. 12.2.1 we start with *hierarchical cluster analysis* (HCA), given the common usage of this unsupervised learning approach in the biomedical community. We then point out that Bayesian approaches can have certain advantages over HCA. We consider *variational* approaches to Bayesian unsupervised learning and illustrate the use of this approach in finding novel disease subtypes in cancer research. We then consider *Markov chain Monte Carlo* (MCMC) approaches, which can be more accurate than variational methods, and illustrate their use in cancer research for finding the most probable pathway structure from a set of candidate pathway topologies.

12.1 Supervised Learning

Many bioinformatics problems involve prediction over two classes; For example, we may want to predict whether a tumor is benign or malignant, based on genetic data. An abstract *learning machine* will learn from *training data* and attempt to *generalize* and thus make predictions on novel input data. For the training data we have a set of input vectors, denoted $\boldsymbol{x}_i$, with each input vector having a number of component *features*. These input vectors are paired with corresponding *labels*, which we denote y_i, and there are m such pairs ($i = 1, \ldots, m$). Thus, for our cancer example, $y_i = +1$ may denote malignant and $y_i = -1$ benign. The matching $\boldsymbol{x}_i$ are input vectors encoding the genetic data derived from each patient i. Typically, we would be interested in quantifying the prediction performance before any practical usage, and so we would evaluate a *test error* based on a *test set* of data.

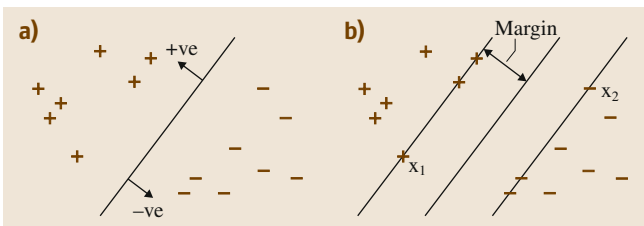

Fig. 12.1 **(a)** The argument inside the decision function of our SVM classifier is $\boldsymbol{w}\cdot\boldsymbol{x}+b$. The separating hyperplane corresponding to $\boldsymbol{w}\cdot\boldsymbol{x}+b=0$ is shown as a line on this plot. This hyperplane separates the two classes of data, with points on one side labeled $y_i = +1$ ($\boldsymbol{w}\cdot\boldsymbol{x}+b \geq 0$) and points on the other side labeled $y_i = -1$ ($\boldsymbol{w}\cdot\boldsymbol{x}+b < 0$). **(b)** The perpendicular distance between the separating hyperplane and a hyperplane through the closest points (the support vectors) is called the *margin*, γ. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are examples of *support vectors* of opposite sign. The hyperplanes passing through the support vectors are *canonical hyperplanes*, and the region between the canonical hyperplanes is the *margin band*. The projection of the vector $(\boldsymbol{x}_1 - \boldsymbol{x}_2)$ onto the normal to the separating hyperplane $(\boldsymbol{w}/\,||\boldsymbol{w}||_2)$ is 2γ

The training data can be viewed as labeled datapoints in an input space, which we depict in Fig. 12.1. For two classes of well-separated data, the learning task amounts to finding a *directed hyperplane*, i.e., an oriented hyperplane such that datapoints on one side will be labeled $y_i = +1$ and those on the other side as $y_i = -1$. The directed hyperplane found by a *support vector machine* is intuitive: it is that hyperplane which is maximally distant from the two classes of labeled datapoints. The closest such points on both sides have most influence on the position of this separating hyperplane and are therefore called *support vectors*. The separating hyperplane is given as $\boldsymbol{w}\cdot\boldsymbol{x}+b=0$ (where "$\cdot$" denotes the inner or scalar product). b is the *bias* or offset of the hyperplane from the origin in input space, and $\boldsymbol{x}$ are points located within the hyperplane. The normal to the hyperplane, the *weight vector* $\boldsymbol{w}$, determines its orientation.

Of course this picture is too simple for many applications. The two clusters could be highly intermeshed with many overlapping datapoints: the dataset is then *not linearly separable*. This situation is one motivation for introducing the concept of *kernels* later in this chapter. We can also see that stray datapoints could have a significant impact on the orientation of the hyperplane, and so we need a mechanism for handling anomalous datapoints and noise.

Statistical learning theory is the theoretical study of learning and generalization. From the perspective of statistical learning theory, the motivation for considering binary classifier SVMs comes from a theoretical upper bound on the *generalization error*, that is, the theoretical prediction error when applying the classifier to novel, unseen instances. This generalization error bound has two important features.

1. The bound is minimized by maximizing the *margin*, γ, i. e., the minimal distance between the hyperplane separating the two classes and the closest datapoints to the hyperplane, and
2. The bound does not depend on the dimensionality of the space.

Suppose we consider a binary classification task with datapoints $\boldsymbol{x}_i$ $(i = 1, \ldots, m)$ having corresponding labels $y_i = \pm 1$ and a *decision function*

$$f(\boldsymbol{x}) = \text{sign}\,(\boldsymbol{w} \cdot \boldsymbol{x} + b)\,, \tag{12.1}$$

where $\cdot$ is the inner product. From the decision function we see that the data is correctly learnt if $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) > 0 \forall i$, since $(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)$ should be positive when $y_i = +1$ and it should be negative when $y_i = -1$. The decision function is invariant under a positive rescaling of the argument inside the sign-function, leading to an ambiguity in defining a distance measure and therefore the margin. Thus, we implicitly define a scale for the $(\boldsymbol{w}, b)$ by setting $\boldsymbol{w} \cdot \boldsymbol{x} + b = 1$ for the closest points on one side and $\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$ for the closest on the other side. The hyperplanes passing through $\boldsymbol{w} \cdot \boldsymbol{x} + b = 1$ and $\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$ are called *canonical hyperplanes*, and the region between these canonical hyperplanes is called the *margin band*. Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be two points inside the canonical hyperplanes on both sides of the separating hyperplane (Fig. 12.1b). If $\boldsymbol{w} \cdot \boldsymbol{x}_1 + b = 1$ and $\boldsymbol{w} \cdot \boldsymbol{x}_2 + b = -1$, we deduce that $\boldsymbol{w} \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 2$. For the separating hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$, the normal vector is $\boldsymbol{w}/\,||\boldsymbol{w}||_2$ (where $||\boldsymbol{w}||_2$ is the square root of $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$). Thus, the distance between the two canonical hyperplanes is equal to the projection of $\boldsymbol{x}_1 - \boldsymbol{x}_2$ onto the normal vector $\boldsymbol{w}/\,||\boldsymbol{w}||_2$, which gives $(\boldsymbol{x}_1 - \boldsymbol{x}_2) \cdot \boldsymbol{w}/\,||\boldsymbol{w}||_2 = 2/\,||\boldsymbol{w}||_2$. As half the distance between the two canonical hyperplanes, the margin is therefore $\gamma = 1/\,||\boldsymbol{w}||_2$. Maximizing the margin is therefore equivalent to minimizing

$$\frac{1}{2}\,||\boldsymbol{w}||_2^2\,, \tag{12.2}$$

subject to the constraints

$$y_i\,(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \quad \forall i\,. \tag{12.3}$$

This is a constrained optimization problem in which we minimize an *objective function* (12.2) subject to the *constraints* (12.3).

As a constrained optimization problem, the above formulation can be reduced to minimization of a *Lagrange function*, consisting of the sum of the objective function and the m constraints multiplied by their respective *Lagrange multipliers*, denoted α_i. We will call this the *primal* formulation

$$L(\boldsymbol{w}, b) = \frac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) - \sum_{i=1}^{m} \alpha_i \left[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 \right]\,, \tag{12.4}$$

where α_i are Lagrange multipliers, and thus $\alpha_i \geq 0$ (a necessary condition for the Lagrange multiplier). At the optimum, we can take the derivatives with respect to b and $\boldsymbol{w}$ and set these to zero

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0\,, \tag{12.5}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i = 0\,. \tag{12.6}$$

Substituting $\boldsymbol{w}$ from (12.6) back into $L(\boldsymbol{w}, b)$ we then get a *dual formulation* (also known as a *Wolfe dual*)

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j\right)\,, \tag{12.7}$$

which must be *maximized* with respect to the α_i subject to the constraints

$$\alpha_i \geq 0, \quad \sum_{i=1}^{m} \alpha_i y_i = 0\,. \tag{12.8}$$

The objective function in (12.7) is *quadratic* in the parameters α_i, and thus it is a constrained *quadratic programming* (QP) problem. QP is a standard problem in optimization theory, so there are a number of resources available, such as *QUADPROG* in *MATLAB*, *MINOS*, and *LOQO*. In addition there are a number of packages specifically written for SVMs such as *SVMlight*, *LIBSVM*, and *SimpleSVM* [12.1].

So far we have not considered point (2), i.e., that the generalization bound does not depend on the dimensionality of the space. For the objective (12.7) we notice that the data $\boldsymbol{x}_i$ only appear inside an inner product. To get an alternative representation of the data, we could therefore map datapoints into a space with different dimensionality, called *feature space*, through

$$\boldsymbol{x}_i \cdot \boldsymbol{x}_j \rightarrow \Phi\,(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)\,, \tag{12.9}$$

where $\Phi(\cdot)$ is the mapping function. Data which are not separable in input space can always be separated in a space of high enough dimensionality. A consequence of the generalization bound, given in (2), is that there is no loss of generalization performance if we map to a feature space where the data are separable and a margin can be defined.

Surprisingly, the functional form of the mapping $\Phi(\boldsymbol{x}_i)$ *does not need to be known in general*, since it is implicitly defined by the choice of the *kernel*: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\mathbf{x}_j)$ or inner product in feature space. For nonseparable continuous-valued data a common choice is the *Gaussian kernel* (Sect. 12.1.3)

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{e}^{-(\boldsymbol{x}_i - \boldsymbol{x}_j)^2/2\sigma^2} \,. \tag{12.10}$$

The introduction of a kernel with its implied mapping to feature space is known as *kernel substitution*. The class of mathematical functions which can be used as kernels is very general. Apart from continuous-valued data we can also consider many data objects which appear in bioinformatics such as graphs (representing *networks* and *pathways*), strings, and sequences (such as genetic or protein sequence data).

For binary classification with a given choice of kernel the learning task therefore involves maximization of

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \,, \tag{12.11}$$

subject to the constraints (12.8). The bias, b, is found separately. For a datapoint with $y_i = +1$,

$$\min_{\{i|y_i=+1\}} [\boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i) + b] = \min_{\{i|y_i=+1\}} \left[\sum_{j=1}^{m} \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right] + b = 1 \,, \tag{12.12}$$

using (12.6), with a similar expression for datapoints labeled $y_i = -1$. We thus deduce that

$$b = -\frac{1}{2} \left\{ \max_{\{i|y_i=-1\}} \left[\sum_{j=1}^{m} \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right] + \min_{\{i|y_i=+1\}} \left[\sum_{j=1}^{m} \alpha_j y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right] \right\} . \tag{12.13}$$

Thus, to construct an SVM binary classifier, we place the data $(\boldsymbol{x}_i, y_i)$ into (12.11) and maximize $W(\alpha)$ subject to the constraints (12.8). From the optimal values of α_i, which we denote $\alpha_i^\star$, we calculate the bias b from (12.13). For a novel input vector $\boldsymbol{z}$, the predicted class is then based on the sign of

$$\phi(\boldsymbol{z}) = \sum_{i=1}^{m} \alpha_i^\star y_i K(\boldsymbol{x}_i, \boldsymbol{z}) + b^\star \,, \tag{12.14}$$

where $b^\star$ denotes the value of the bias at optimality.

12.1.1 Multiclass Classification and Other Extensions

Multiclass Classification

Many bioinformatics problems involve multiclass classification, and a number of schemes have been outlined. If the number of classes is small then we can use a *directed acyclic graph* (DAG) [12.2] with the learning task reduced to binary classification at each node. The idea is illustrated in Fig. 12.2. Suppose we consider a three-class classification problem. The first node is a classifier making the binary decision label 1 versus label 3, say. Depending on the outcome of this decision, the next steps are the decisions 1 versus 2, or 2 versus 3. We could also use a series of one-against-all classifiers [12.3]. Thus, we construct C separate SVMs, with the c-th SVM trained using data from class c as the positively labeled samples and the remaining classes as the negatively labeled samples. Associated with the c-th SVM we have $f_c(\boldsymbol{z}) = \sum_i y_i^c \alpha_i^c K(\boldsymbol{z}, \boldsymbol{x}_i) + b^c$, and the novel input $\boldsymbol{z}$ is assigned to class c such that $f_c(\boldsymbol{z})$ is largest. Other schemes have been suggested [12.4–7].

Learning with Noise: Soft Margins

Many biomedical datasets are intrinsically noisy, and a learning machine could fit to this noise, leading to poor generalization. As remarked earlier, outliers can have an undue influence on the position of the separating hyperplane used by an SVM (Fig. 12.1). Potential

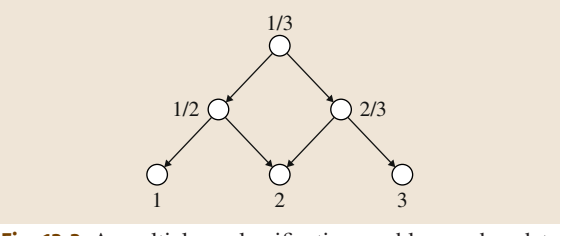

Fig. 12.2 A multiclass classification problem reduced to a series of binary classification tasks

noise in a dataset can be handled by the introduction of a *soft margin* [12.8]. Two schemes are commonly used. With an L_1 *error norm*, the learning task is the same as in (12.11, 12.8) except for the introduction of the *box constraint*

$$0 \le \alpha_i \le C \,. \tag{12.15}$$

On the other hand, for an L_2 *error norm*, the learning task is (12.11, 12.8) except for the addition of a small positive constant to the leading diagonal of the kernel matrix

$$K(\boldsymbol{x}_i, \boldsymbol{x}_i) \leftarrow K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \lambda \,. \tag{12.16}$$

The appropriate values of these parameters can be found by means of a *validation study*. With sufficient data we would split the dataset into a *training set*, a *validation set*, and a *test set*. With regularly spaced values of C or λ, we train the SVM on the training data and find the best choice for this parameter based on the validation error. With more limited data, we may use *cross-validation*, or rotation estimation, in which the data are randomly partitioned into subsets and rotated successively as training and validation data.

With many biomedical datasets there is an imbalance between the amount of data in different classes, or the significance of the data in the two classes can be quite different; For example, for the detection of tumors on magnetic resonance imaging (MRI) scans, it may be best to allow a higher number of false positives if this improved the true-positive detection rate. The balance between the detection rate for different classes can be easily shifted by introducing *asymmetric soft margin parameters* [12.9]. Thus, for binary classification with an L_1 error norm we use $0 \le \alpha_i \le C_+$ $(y_i = +1)$ and $0 \le \alpha_i \le C_-$ $(y_i = -1)$, but $K(\boldsymbol{x}_i, \boldsymbol{x}_i) \leftarrow K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \lambda_+$ (if $y_i = +1$) and $K(\boldsymbol{x}_i, \boldsymbol{x}_i) \leftarrow K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \lambda_-$ (if $y_i = -1$) for the L_2 error norm.

Introducing a Confidence Measure

Suppose we are using a support vector machine for diagnostic categorization or prediction of disease progression; then, it would be plainly useful to have a confidence measure associated with the class assignment. A clinician could be expected to plan differently with a high confidence prediction over a low confidence one. A SVM has an in-built quantity that could provide a confidence measure for the class assignment, i. e., the distance of a new point from the separating hyperplane (Fig. 12.1). A new datapoint with a large distance from the separating hyperplane should be assigned a higher degree of confidence than a point which lies close to the hyperplane. Before thresholding, the output of a SVM is given by

$$\phi(z) = \sum_i y_i \alpha_i K(\boldsymbol{x}_i, z) + b \,. \tag{12.17}$$

One approach is to fit a *probability* measure $p(y|\phi)$ directly [12.10]. A good choice for the mapping function is the *sigmoid*

$$p(y = +1|\phi) = \frac{1}{1 + \exp(A\phi + B)} \,, \tag{12.18}$$

with the parameters A and B found from the training set $(y_i, \phi(\boldsymbol{x}_i))$. Let us define t_i as the target probabilities

$$t_i = \frac{y_i + 1}{2} \,, \tag{12.19}$$

so for $y_i \in \{-1, 1\}$ we have $t_i \in \{0, 1\}$. We find A and B by performing the following minimization over the entire training set:

$$\min_{A,B} \left[-\sum_i t_i \log(p_i) + (1 - t_i) \log(1 - p_i) \right] \,, \tag{12.20}$$

where p_i is simply (12.18) evaluated at $\phi(\boldsymbol{x}_i)$. This is a straightforward two-dimensional minimization problem which can be solved using a variety of optimization methods. Once the sigmoid has been found using the training set, we can use (12.18) to calculate the probability that a new test point belongs to either class. Figure 12.3 shows the margins of training data points

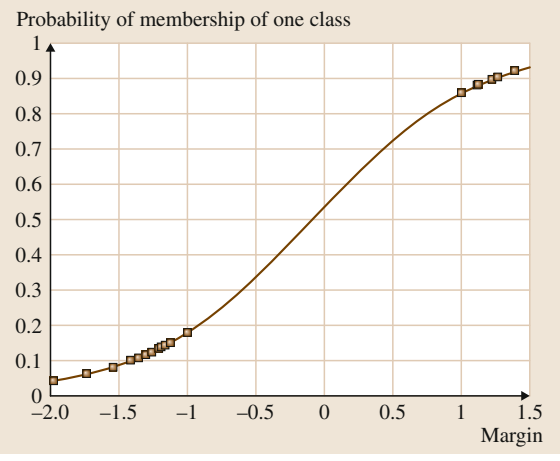

Fig. 12.3 Probability of membership of one class (y-axis) versus margin (x-axis). The plot shows the training points and fitted sigmoid for an ovarian cancer dataset

(x-axis) and fitted sigmoid for a microarray dataset for ovarian cancer. The distinction is ovarian cancer versus normal. There are no datapoints present in a band between +1 and −1 since this corresponds to the margin band of the SVM.

Regression

Some bioinformatics applications involve *regression* and the construction of models with real-valued labels y_i. To model the dependency between the input vectors $\boldsymbol{x}_i$ and the y_i we could use a linear function of the form

$$g(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i \tag{12.21}$$

to approximate y_i. Thus, we could minimize the following function in $\boldsymbol{w}$:

$$L(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{m}\left[y_i - g(\boldsymbol{x}_i)\right]^2 \tag{12.22}$$

to get a solution $y_i \approx g(\boldsymbol{x}_i)$. If we make a mapping to feature space $\boldsymbol{x}_i \rightarrow \Phi(\boldsymbol{x}_i)$ and introduce a variable $\xi_i = y_i - \boldsymbol{w}^{\mathrm{T}}\Phi(\boldsymbol{x}_i)$, then (12.22) could be reformulated as

$$\min_{\boldsymbol{w},\boldsymbol{\xi}}\left\{L = \sum_{i=1}^{m}\xi_i^2\right\}\,, \tag{12.23}$$

subject to the constraints

$$y_i - \boldsymbol{w}^{\mathrm{T}}\Phi(\boldsymbol{x}_i) = \xi_i \forall i \tag{12.24}$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \le B^2\,. \tag{12.25}$$

The latter constraint (12.25) is a *regularization* constraint on the $\boldsymbol{w}$; that is, it is used to avoid an overcomplex solution which fits to noise in the data, leading to poor generalization. As a constrained optimization problem with objective function (12.23) and two constraint conditions (12.24) and (12.25), we derive a Lagrange function

$$L = \sum_{i=1}^{m}\xi_i^2 + \sum_{i=1}^{m}\beta_i\left[y_i - \boldsymbol{w}^{\mathrm{T}}\Phi(\boldsymbol{x}_i) - \xi_i\right] + \lambda\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - B^2\right) \tag{12.26}$$

with Lagrange multipliers β_i and λ for these two constraint conditions. If we take derivatives of L with respect to ξ_i and $\boldsymbol{w}$, we get

$$\xi_i = \frac{1}{2}\beta_i\,, \tag{12.27}$$

$$\boldsymbol{w} = \frac{1}{2\lambda}\sum_{i=1}^{m}\beta_i\Phi(\boldsymbol{x}_i)\,. \tag{12.28}$$

Substituting these back into L gives the *dual* formulation

$$W = \sum_{i=1}^{m}\left(-\frac{1}{4}\beta_i^2 + \beta_i y_i\right) - \frac{1}{4\lambda}\sum_{i,j=1}^{m}\left(\beta_i\beta_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)\right) - \lambda B^2\,, \tag{12.29}$$

which with a redefined variable $\alpha_i = \beta_i/2\lambda$ (a positive rescaling, since $\lambda \ge 0$) gives the following restatement:

$$\max_{\alpha_i,\lambda}\left\{W = -\lambda^2\sum_{i=1}^{m}\alpha_i^2 + 2\lambda\sum_{i=1}^{m}\alpha_i y_i - \lambda\sum_{i,j=1}^{m}\alpha_i\alpha_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) - \lambda B^2\right\}\,. \tag{12.30}$$

In contrast to a SVM for binary classification, where we must solve a constrained QP problem, (12.30) gives a direct solution

$$\alpha = (\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}\,. \tag{12.31}$$

This gives a means for finding $\boldsymbol{w}$ in (12.21) via (12.28). The above is one of the simplest kernel-based approaches to regression [12.11, 12]. However, for prediction on a novel datapoint we implicitly use *all* the datapoints via the kernel matrix $\boldsymbol{K}$. Sample sparsity is desirable since it reduces the complexity of the model. For this reason, it not the best approach to finding a regression function, and thus a number of approaches which involve constrained quadratic programming have been developed [12.13–17]. These minimize the number of support vectors favoring sparse hypotheses and giving smooth functional approximations to the data.

12.1.2 Case Study 1: Predicting Disease Progression

As an example of the use of SVMs within the context of bioinformatics, we consider an application to predicting disease progression. In this example, the objective is to predict relapse versus nonrelapse for Wilm's tumor, a cancer which affects children and young adults [12.18]. This tumor originates in the kidney, but it is a curable disease in the large majority of affected children. However, there is a recognized aggressive subtype with a high probability of relapse within a few years. It is therefore clini-

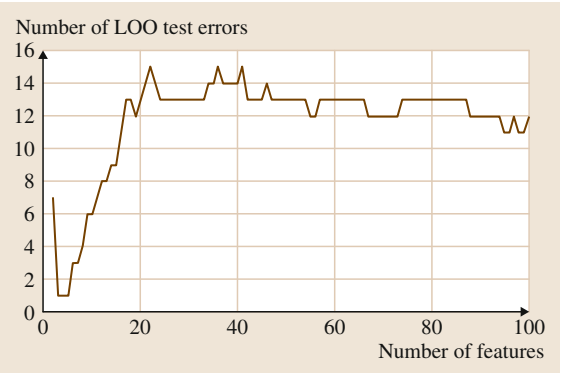

Fig. 12.4 The number of LOO test errors (y-axis) versus number of top-ranked features (x-axis) remaining (with features ranked by a t-test statistic) for predicting relapse or nonrelapse for Wilm's tumor

cally important to predict risk of relapse when the disease is first diagnosed, with an alternative treatment regime if risk of relapse is high. In this study we used microarray data as input to the support vector machine. The microarray had 30 720 probes, each measuring *gene expression*, roughly quantifying the amount of protein produced per gene. A number of these readings were poor quality, so quality filtering was used, reducing the number of *features* to 17 790. The dataset consisted of 27 samples, of which 13 samples were from patients who relapsed with the disease within 2 years and the remainder labeled as nonrelapse due to long-term survival without disease recurrence.

The task is binary classification, and the large number of features means the relatively small number of datapoints are embedded in a high-dimensional space. As a consequence, the dataset is separable and we use a *linear kernel* $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$. Since we do not have large training and test sets we use *leave-one-out* (LOO) testing; i. e., we train on 26 samples and evaluate classifier performance on the single left-out datapoint, successively rotating the test datapoint through the data. The vast majority of features are likely to be irrelevant, so we use *feature selection* to remove uninformative features and thus improve performance.

Using a t-test statistic to rank features [12.19] we can obtain a minimal LOO test error of 1 from 27 (the t-test must be run separately per LOO partitioning to avoid corrupting the test statistic). However, this tentative prediction performance is only achieved with a small set of the most informative features, and, if all features are included, the vast number of uninformative and noisy features is sufficient to overwhelm this predictive genetic signature.

12.1.3 Different Types of Kernels

Biomedical data can appear in many different formats, and in this section we consider how to construct kernels representing these different types of data.

Permissable Kernels

If a proposed kernel matrix is *positive semidefinite* (PSD) then it is an allowable kernel. For any arbitrary set of real-valued variables $a_1, \ldots, a_m$, a PSD kernel satisfies

$$\sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0. \tag{12.32}$$

This type of kernel is symmetric, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, with positive components on the diagonal, $K(\boldsymbol{x}, \boldsymbol{x}) \geq 0$. An example is the *linear kernel* introduced earlier

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j \,. \tag{12.33}$$

It is symmetric, $K(\boldsymbol{x}, \boldsymbol{x}) \geq 0$, and it satisfies (12.32) since

$$\sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) = \left|\left| \sum_{i=1}^{m} a_i \boldsymbol{x}_i \right|\right|^2 \geq 0 \,. \tag{12.34}$$

We can determine if a proposed kernel matrix is PSD by determining its spectrum of eigenvalues: if the matrix has at least one negative eigenvalue λ with corresponding eigenvector $\boldsymbol{v}$, say, then $\boldsymbol{v}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{v} = \lambda \boldsymbol{v}^{\mathrm{T}} \boldsymbol{v} < 0$, so it is not PSD. There are, indeed, strategies for handling non-PSD kernel matrices [12.20–24].

From permissable kernels we can construct other permissable kernels; For example,

1. If $K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel then so is
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = c K_1(\boldsymbol{x}_i, \boldsymbol{x}_j) \,, \tag{12.35}$$
where c is a positive constant.
2. If $K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $K_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are two kernels then the sum $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + K_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and the product $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K_1(\boldsymbol{x}_i, \boldsymbol{x}_j) K_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are both permissable kernels.
3. If $K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel and $f(\boldsymbol{x})$ is any function of $\boldsymbol{x}$, then the following is a permissable kernel:
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = f(\boldsymbol{x}_i) K_1(\boldsymbol{x}_i, \boldsymbol{x}_j) f(\boldsymbol{x}_j) \,. \tag{12.36}$$
4. If $K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel then so is
$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = p\left[K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\right], \tag{12.37}$$

where $p(\cdot)$ is a polynomial with nonnegative coefficients. As an example, the following is a kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp[K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)] \,, \tag{12.38}$$

since $\exp(\cdot)$ can be expanded in a Taylor series with positive coefficients.

A further manipulation we can apply to a kernel matrix is *normalization*. This is achieved using a modified mapping function to feature space: $\boldsymbol{x} \to \Phi(\boldsymbol{x}) / ||\Phi(\boldsymbol{x})||_2$. The normalized kernel is then

$$\begin{aligned}
\widehat{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \frac{\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)}{||\Phi(\boldsymbol{x}_i)||_2 \left|\left|\Phi(\boldsymbol{x}_j)\right|\right|_2} \\
&= \frac{\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)}{\sqrt{\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_i)} \sqrt{\Phi(\boldsymbol{x}_j) \cdot \Phi(\boldsymbol{x}_j)}} \\
&= \frac{K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i) K(\boldsymbol{x}_j, \boldsymbol{x}_j)}} \,.
\end{aligned} \tag{12.39}$$

Thus, consider the Gaussian kernel introduced in (12.10). Since

$$\begin{aligned}
K(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^2}{2\sigma^2}\right) \\
&= \exp\left(\frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\sigma^2} - \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_i}{2\sigma^2} - \frac{\boldsymbol{x}_j \cdot \boldsymbol{x}_j}{2\sigma^2}\right) \\
&= \frac{\exp\left(\frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\sigma^2}\right)}{\sqrt{\exp\left(\frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_i}{\sigma^2}\right) \exp\left(\frac{\boldsymbol{x}_j \cdot \boldsymbol{x}_j}{\sigma^2}\right)}} \,,
\end{aligned} \tag{12.40}$$

it is a normalized kernel. Since we know that the linear kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$ is a permissable kernel, validity of a Gaussian kernel follows from properties (1), (2), and (4) above.

For the Gaussian kernel, and various other kernels, there is a *kernel parameter* (the σ for the Gaussian). The value for this parameter needs to be found, and there are several ways to do this. If we have enough data we can split it into a *training set*, a *validation set*, and a *test set*. We then pursue a *validation* study in which the learning machine is trained at regularly spaced choices of the kernel parameter, and we use that value which minimizes the validation error (the error on the validation data). If insufficient data are available and we are considering classification, then we can estimate the kernel parameter using generalization bounds with no recourse to using validation data [12.25–29].

Kernels for Strings and Sequences

Strings appear in many bioinformatics contexts; For example, we could be considering DNA genetic sequences composed of the four DNA bases A, C, G, and T, or we could be considering proteinogenic amino acid sequences composed of the 21 amino acids found in eukaryotes. Strings can be defined as ordered sets of symbols drawn from an alphabet. We can evidently see a degree of similarity between strings. Thus, suppose we consider the genetic sequences ACTGA, CCACTG, and CTGACT. They have the string CTG in common, and a matching algorithm should pick up this similarity irrespective of the differing prefixes and suffixes. Strings could differ by *deletions* or *insertions*, thus ACGA differs from ACTGA by a gap consisting of a single deletion.

We can consider two distinct categories when matching ordered sets of symbols [12.30]. The first we will call *string matching*: in this case contiguity of the symbols is important. For the second category, *sequence matching*, only the order is important. Thus, for our example with ACGA and ACTGA, there are only two short contiguous strings in common: AC and GA. On the other hand, A, C, G, and A are ordered the same way in both words. When matching the same genes between two individuals, there will be extensive commonality, interrupted by occasional mutations and rare deletions and insertions. In this section we consider the p-spectrum kernel (contiguity is necessary), the subsequence kernel (contiguity is not necessary and the order of symbols is important), and a gap-weighted kernel which is influenced by both contiguity and order.

The p-Spectrum Kernel. Two strings can be compared by counting the number of contiguous substrings of length p which are in common. A p-spectrum kernel [12.31–33] is based on the set of frequencies of all *contiguous substrings of length p*; For example, suppose we wish to compute the 2-spectrum of the string $s = \text{CTG}$. There are two contiguous substrings of length $p = 2$, namely $u_1 = \text{CT}$ and $u_2 = \text{TG}$, both with frequency of 1. As a further example, let us consider the set of strings $s_1 = \text{CTG}$, $s_2 = \text{ACT}$, $s_3 = \text{CTA}$, and $s_4 = \text{ATA}$. The 2-spectrum mapping function Φ is given in Table 12.1, where each entry is the number

Table 12.1 A mapping function Φ for the p-spectrum kernel

Φ	CT	AT	TG	TA
CTG	1	0	1	0
ATG	0	1	1	0
CTA	1	0	0	1
ATA	0	1	0	1

Table 12.2 The p-spectrum kernel matrix from the mapping function in Table 12.1

K	CTG	ATG	CTA	ATA
CTG	2	1	1	0
ATG	1	2	0	1
CTA	1	0	2	1
ATA	0	1	1	2

of occurrences of the substring u (say $u = \text{CT}$) in the given string (say $s = \text{CTG}$). The corresponding kernel is shown in Table 12.2.

Thus, to compute the (CTG, ATG) entry in the kernel matrix, we sum the products of the corresponding row entries under each column in the mapping function (Table 12.1). Only the pair of entries in the TG substring column both have nonzero entries of 1, giving $K(\text{CTG}, \text{ATG}) = 1$. For an entry on the diagonal of the kernel matrix, we take the sum of the squares of the entries in the corresponding row in Table 12.1. Thus, for $K(\text{CTG}, \text{CTG})$, there are nonzero entries of 1 under CT and TG, and so $K(\text{CTG}, \text{CTG}) = 2$.

The All-Subsequence Kernel. With this kernel the implicit mapping function is taken *over all contiguous and noncontiguous ordered subsequences of a string*, which includes the empty set. As an example let us consider two sequences $s_1 = \text{CTG}$ and $s_2 = \text{ATG}$. Let Ω represent the empty set, then the mapping function is given in Table 12.3.

The off-diagonal terms in the kernel matrix are then evaluated as the sum across all columns of the products of the two entries in each column. The diagonal terms are the sum of squares of all entries in a row (Table 12.4).

Finally, we can consider a *gap-weighted subsequence kernel*. With this kernel a penalization is used so that the length of the intervening gap or insertion decreases the score for the match. As an example, CTG is a subsequence of CATG and CAAAAATG. However, CTG differs from CATG by one deletion, but CTG differs from CAAAAATG by a gap of five symbol deletions. By appropriately weighting the penalty associated with the gap or insertion, we can interpolate between a p-spectrum kernel and the all-subsequence kernel.

Table 12.4 The all-subsequences kernel matrix for the mapping function in Table 12.3

K	CTG	ATG
CTG	8	4
ATG	4	8

Kernels for Graphs

Graphs appear in many settings in bioinformatics; For example, we can use a graph to represent a transcriptional regulatory network with the genes as the nodes and the edges representing functional connections between genes. We can consider two types of similarity. For a given graph we may be interested in the similarity of two nodes within the same graph. On the other hand we may be interested in constructing a measure of similarity between two different graphs. We can construct kernels for both cases, and we refer to *kernels on graphs* [12.34, 35] when constructing a within-graph kernel between nodes and *kernels between graphs* [12.36, 37] for the latter comparison.

12.1.4 Multiple Kernel Learning

Many bioinformatics prediction tasks involve different types of data. In the preceding sections we saw that kernels are available for encoding a variety of different data types. Thus, we now consider prediction based on multiple types of input data. Plainly, if we build a predictor which can use all available relevant information, then it is likely to be more accurate than a predictor based on one type of data only.

As an example, for our case study 2, considered shortly, we predict protein fold class based on the use of 12 different types of data. This potentially includes sequence data derived from RNA sequences but also continuous-valued data from various physical measurements. Later, in case study 5, we consider *network completion*. Based on a training set of known links and nonlinks we attempt to predict possible links to new nodes in an incomplete network. For the case of network completion with protein–protein interaction data there are a variety of informative data types such as gene expression correlation, protein cellular localization, and phylogenetic profile data. Individually, these

Table 12.3 A mapping function Φ for the all-subsequences kernel

Φ	Ω	A	C	G	T	CT	AT	TG	CG	AG	CTG	ATG
CTG	1	0	1	1	1	1	0	1	1	0	1	0
ATG	1	1	0	1	1	0	1	1	0	1	0	1

types of data may be weakly informative for the purposes of prediction. However, taken together, they may yield a stronger predictive signal.

This type of problem is called *multiple kernel learning* (MKL). The most common approach to MKL is to use a linear combination of candidate kernels, with these kernels representing different types of input data. Let $\mathcal{K}$ be such a set of candidate kernels, then the objective of MKL is to simultaneously find the optimal weighted combination of these kernels *and* the best classification function. With such a linear combination of p prescribed kernels $\{K_\ell : \ell = 1, \ldots, p\}$, we have a composite kernel

$$K = \sum_{\ell=1}^{p} \lambda_\ell K_\ell \,, \tag{12.41}$$

where $\sum_{\ell=1}^{p} \lambda_\ell = 1$, $\lambda_\ell \geq 0$, and the λ_ℓ are called the *kernel coefficients*.

There are a number of criteria for learning the kernel. For SVM binary classification an appropriate criterion would be to maximize the margin with respect to all the kernel spaces capable of discriminating different classes of labeled data. In particular, for a given kernel $K \in \mathcal{K}$ with $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$, we have seen that maximizing the margin involves minimizing $\|\boldsymbol{w}\|^2$ in feature space subject to

$$y_i \left(\boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i) + b\right) \geq 1, \quad i = 1, \ldots, m \,. \tag{12.42}$$

As we have seen, maximizing the margin subject to these constraints gives the following dual problem:

$$\omega(K) = \max_{\alpha} \left\{ \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) : \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0 \right\} . \tag{12.43}$$

If $\mathcal{K}$ is now a linear combination of kernel matrices, this maximum margin approach to kernel combination learning reduces to

$$\min_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}} \left\{ \mathcal{L}(\alpha, \lambda) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left[\sum_{\ell=1}^{p} \lambda_\ell K_\ell(x_i, x_j) \right] \right\} \tag{12.44}$$

subject to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C \,, \tag{12.45}$$

$$\sum_{\ell=1}^{p} \lambda_\ell = 1 \quad \lambda_\ell \geq 0 \,. \tag{12.46}$$

$\mathcal{L}(\alpha, \lambda)$ is *concave* with respect to $\boldsymbol{\alpha}$ and *convex* with respect to $\boldsymbol{\lambda}$. A number of approaches have been proposed for dealing with this type of optimization problem. One of the earliest approaches [12.38] proposed a *semidefinite programming* (SDP) approach. SDP is computationally intensive, so more efficient methods were developed subsequently [12.38–43].

12.1.5 Case Study 2: Protein Fold Prediction Using Multiple Kernel Learning

During *folding* a protein forms its final three-dimensional structure. Understanding a protein's structure gives important insights into function. Its structure can lead to an understanding of protein–protein interaction or likely biological function. A knowledge of protein structure is important in the design of small molecular inhibitors for disabling the function of target proteins. We can use machine learning methods to predict protein structure based on sequence and other types of data: indeed this is an obvious application domain for MKL techniques. In this case study we only consider a subproblem of structure prediction in which the predicted label is over a set of *fold classes*. The fold classes are a set of structural components, common across proteins, which give rise to the overall three-dimensional structure. In this study we will use 27 fold classes with 313 proteins used for training and 385 for testing.

There are a number of relevant types of data which can be used to predict fold class, and in this study we use 12 different types of data, each encoded into a kernel. These data types included sequence and physical measurements such as hydrophobicity, polarity, and van der Waals volume. In Fig. 12.5 we illustrate the performance of a MKL algorithm on this dataset. In Fig. 12.5a the vertical bars indicate the test set accuracy based on using one type of data only: for example, H is hydrophobicity, P is polarity, and V is van der Waals volume. The horizontal line indicates the performance of the MKL algorithm with all data types included. Thus, we get an improvement in per-

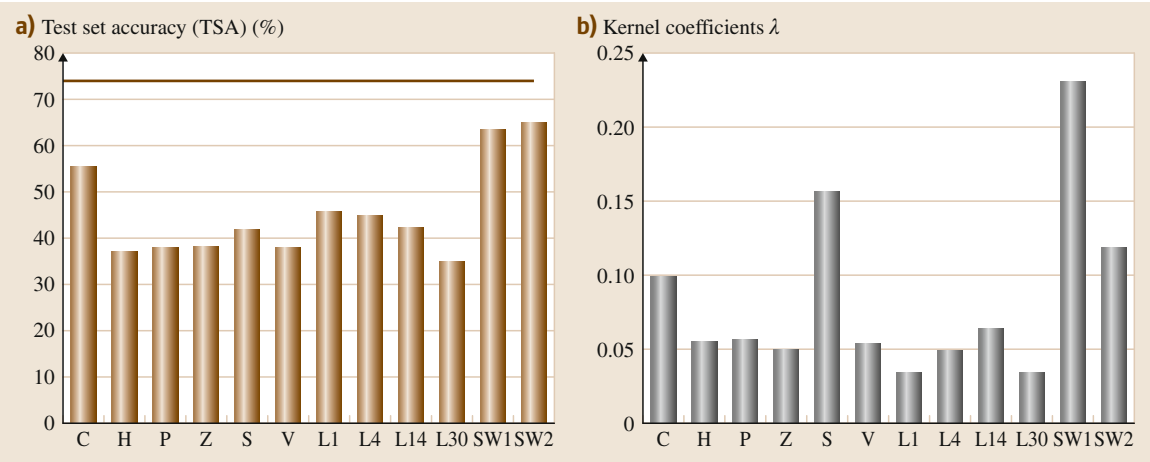

Fig. 12.5a,b Performance of a MKL method [12.43] on a protein fold prediction dataset. There are 27 classes and 12 types of data. **(a)** Test set accuracy (TSA, in %) based on individual data types (*vertical bars*) and using MKL (*horizontal line*). **(b)** Kernel coefficients λ_ℓ, which indicate the relative significance of individual types of data

formance if we use all available relevant sources of data over just using the single most informative data source.

Figure 12.5b gives the values of the kernel coefficients λ_ℓ based on using a linear combination (12.41). The relative height of the peaks indicates the relative significance of different types of input data. This algorithm indicates that all 12 types of data are relevant, though some types of data are more informative than others (the most informative, SW1 and SW2, are based on sequence alignments). MKL methods have been successfully demonstrated on other bioinformatics problems requiring integration of heterogeneous datasets [12.38, 43–46].

12.2 Unsupervised Learning

Having considered supervised learning, we now discuss unsupervised learning and the discovery of structure in biomedical datasets. Hierarchical cluster analysis is a commonly used approach to unsupervised learning in the biomedical literature, and so we briefly review this topic in Sect. 12.2.1. One of our main objectives in this section will be to show that there are some advantages to using more contemporary methods, for example, from Bayesian statistics. In Sects. 12.2.3–12.2.5 we introduce *variational* methods. These are not as accurate as the *Markov chain Monte Carlo* (MCMC) methods discussed in Sect. 12.2.7. However, they are fast in practice and thus suited to some of the large datasets which appear in many bioinformatics applications. After introducing variational methods we consider an application to the interpretation of gene expression array datasets in cancer research. After introducing MCMC, we illustrate its use with network inference in Sect. 12.2.8 with an application to finding the most probable network topology given a set of candidate network topologies.

12.2.1 Hierarchical Cluster Analysis

In the biomedical literature, hierarchical cluster analysis (HCA) is a commonly used technique for unsupervised learning. This approach can be divided into *agglomerative methods*, which proceed through a series of successive fusions of m samples into larger and larger clusters, and *divisive methods*, which systematically separate clusters into smaller and smaller groupings. HCA methods are usually represented by a *dendrogram* which illustrates the successive fusions or divisions produced by this approach. In this section we only consider agglomerative methods. In this case we start with m single sample clusters, representing each data point. At each stage in the clustering procedure we fuse those samples or groups of samples which are

currently closest to each other. There are a number of criteria for evaluating the similarity or closeness of data points. Thus, if x_{id} corresponds to the i-th sample ($i = 1, \ldots, m$) with d the corresponding *feature* index ($d = 1, \ldots, p$), then a commonly used similarity measure is the *squared distance*

$$D(x_i, x_j) = \sum_{d=1}^{p} (x_{id} - x_{jd})^2 \, .$$

Other criteria are used such as the *correlation coefficient*,

$$C(x_i, x_j) = \frac{\sum_d (x_{id} - \overline{x}_i)(x_{jd} - \overline{x}_j)}{\sqrt{\sum_d (x_{id} - \overline{x}_i)^2 \sum_d (x_{jd} - \overline{x}_j)^2}} \, ,$$

where $\overline{x}_i = (\sum_d x_{id})/p$. If each sample vector is standardized to zero mean and unit standard deviation, then clustering based on the correlation coefficient becomes equivalent to use of a squared distance. Using the chosen distance measure, we then derive a *distance matrix* encoding the distances between all data points.

Just as there are a number of ways of quantifying similarity, there are a number of criteria for deciding which clusters to fuse at each stage. Six methods are commonly used in practice: single linkage, complete linkage, average linkage, the centroid and median methods, and Ward's method. We will illustrate the approach with *single linkage clustering* as one of the simplest of these methods. In this case the distance between groupings is defined as the shortest distance between any pair of samples. This method is best illustrated with an example. Thus, suppose we have a set of five samples with a corresponding initial distance matrix given by

$$\boldsymbol{D}_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 4 & 8 & 9 & 7 \\ 4 & 0 & 6 & 5 & 6 \\ 8 & 6 & 0 & 3 & 8 \\ 9 & 5 & 3 & 0 & 2 \\ 7 & 6 & 8 & 2 & 0 \end{pmatrix} \end{array} .$$

In this matrix it is evident that the smallest distance is that between samples 4 and 5. Thus, we place these two samples into a new cluster. The new distances between this cluster, which we label (45) and the other three samples is obtained from the above distance matrix as

$$d_{1(45)} = \min(d_{14}, d_{15}) = d_{15} = 7 \, ,$$
$$d_{2(45)} = \min(d_{24}, d_{25}) = d_{24} = 5 \, ,$$
$$d_{3(45)} = \min(d_{34}, d_{35}) = d_{34} = 3 \, .$$

A new distance matrix can be derived based on these distances and the remaining set of pre-existing distances, thus

$$\boldsymbol{D}_2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ (45) \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & (45) \end{array} \\ \begin{pmatrix} 0 & 4 & 8 & 7 \\ 4 & 0 & 6 & 5 \\ 8 & 6 & 0 & 3 \\ 7 & 5 & 3 & 0 \end{pmatrix} \end{array} .$$

The smallest entry in this new distance matrix is then between sample 3 and the cluster (45), and so we form a new three-member cluster and a new set of distances

$$d_{1(345)} = \min(d_{13}, d_{14}, d_{15}) = d_{15} = 7 \, ,$$
$$d_{2(345)} = \min(d_{23}, d_{24}, d_{25}) = d_{24} = 5 \, ,$$

leading to a new 3×3 distance matrix. This process is iterated until all data points belong to one cluster. The sequence of clusterings is therefore

Step	Clusters
1	(1), (2), (3), (4), (5)
2	(1), (2), (3), (45)
3	(1), (2), (345)
4	(12), (345)
5	(12345)

Of course, using the closest points within each cluster is not necessarily a good fusion criterion. With *complete linkage clustering* the distance between two groupings is defined as the most distant pair of data points, with each pair consisting of one sample from each of the two groups. Again, this type of method can be influenced by outliers within each cluster, and so a better method is to use the average of a cluster instead. With *average clustering* the distance between two groupings is defined as the average of the distances between all pairs of samples, with one sample in a pair from each group. Thus, with our example above, after merging data points 4 and 5 into the first cluster, we would compute the new set of distances from this cluster to the other three data points through

$$d_{1(45)} = \tfrac{1}{2}(d_{14} + d_{15}) = 8.0 \, ,$$
$$d_{2(45)} = \tfrac{1}{2}(d_{24} + d_{25}) = 5.5 \, ,$$
$$d_{3(45)} = \tfrac{1}{2}(d_{34} + d_{35}) = 5.5 \, .$$

With *centroid clustering* each grouping is represented by a mean vector, that is, a vector composed of the mean values of each feature taken over all samples

within the grouping. The distance between two clusters is then the distance between the corresponding mean vectors. However, centroid clustering has the following disadvantage: if the number of members in one cluster is very different from the other, then, when they are fused, the centroid of the new cluster will be most heavily influenced by the larger grouping and may remain within that cluster. Clustering can be made independent of group size by assuming that the two groups are of equal size. *Median clustering* is a variant on centroid clustering in which the cluster arising from the fusion lies at such a midpoint between the two clusters. Finally, with *Ward's method*, the two clusters chosen for fusion are the two which result in the least increase in the sum of the distances of each sample to the centroid of its originating cluster.

Of course, the above methods provide an approach to finding a cluster structure but they do not indicate the most likely number of clusters in the data. However, various criteria have been proposed to indicate the most probable number of clusters [12.48–50].

12.2.2 Case Study 3: An HCA Application to Colon Cancer

As an illustration, we now apply HCA to a gene expression microarray dataset from a cancer study. A DNA microarray has a series of microscopic probes, each constructed from strings of nucleotides. A microarray has tens of thousands of such probes, each of which can hybridize to a particular target strand of complementary DNA (cDNA). Probe–target hybridization can be measured by using fluorophore labels, for example, with altered levels of gene expression quantified by the level of fluorescence. In Fig. 12.6 we depict the corresponding dendrogram (using *average linkage* and a *correlation coefficient* distance measure) for a small microarray dataset consisting of 17 colon tumor and 18 normal samples. The dendrogram has been largely successful in separating normal from cancer samples.

12.2.3 Bayesian Unsupervised Learning

HCA has been the method of choice for cluster analysis for many biomedical publications. However, HCA does have some drawbacks. Firstly, there is an implicit assumption that each sample is associated to a particular cluster. This may not be realistic in many situations where a sample should be better represented as overlapping several clusters. Thus, tumors can be genetically heterogeneous; i. e., tissue regions in different regions of the tumor may have different genetic signatures. The models we describe below are *mixed membership models* with each sample represented as a combinatorial mixture over clusters. With the clinical assignment of patient samples to clusters or disease subtypes it would also be useful to associate a confidence measure with the cluster assignment. This can be achieved using the Bayesian methods outlined here. Further advantages of a Bayesian approach are the use of sound methods to objectively assess the number of sample clusters and the means to penalize overfitting (at the top levels of the dendrogram in Fig. 12.6 we are finding structure, but, toward the leaves at the base, we are starting to fit to noise in the data).

Let us suppose that M is a model and D is the data, then $p(M|D)$ represents the probability of a model given the data. Intuitively we should seek to maximize

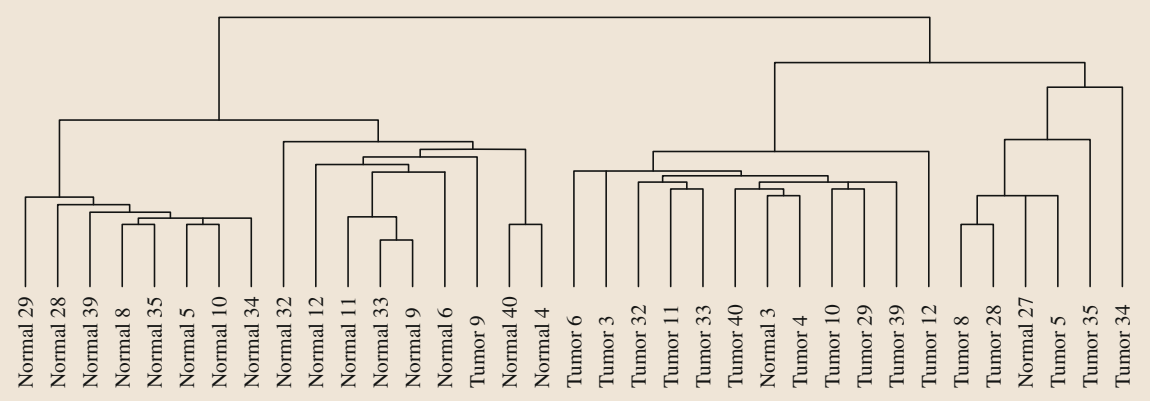

Fig. 12.6 HCA applied to a gene expression microarray dataset [12.47] consisting of 35 samples comprising 17 colon cancer and 18 normal colon samples

the probability of the model given the data. Given an assumed model structure, a fit to data is achieved through the adjustment of *model parameters*, which we denote Θ as a set and represent as the components of a vector $\boldsymbol{\theta}$. *Bayes's theorem* then implies that we should maximize

$$p(\Theta|D) = \frac{p(D|\Theta)p(\Theta)}{p(D)} , \tag{12.47}$$

where $p(\Theta|D)$ is the *posterior*, $p(D|\Theta)$ is the *likelihood*, and $p(\Theta)$ is the *prior*. Thus, $p(\Theta|D) \sim p(D|\Theta)p(\Theta)$ states that multiplying the likelihood by our prior beliefs about the parameters, $p(\Theta)$, will give us posterior beliefs about the parameters, having observed the data, $p(\Theta|D)$. The normalization term $p(D)$ is called the *evidence* and can be found through an integration over the $\boldsymbol{\theta}$ (we will refer to this as *marginalizing* out the $\boldsymbol{\theta}$)

$$p(D) = \int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} . \tag{12.48}$$

Maximizing the probability of a model, given the data, would require finding the optimal set of values for the model parameters, which we denote $\widehat{\Theta}$. We will call this a set of *point estimates* for these parameters.

However, we cannot make inferences which are not justified by the data. As a consequence, given some set of data, the most we can really say is that there is a spectrum of models which fit the data and some of these models are more probable than others. We call this a *posterior distribution over models*. This posterior distribution carries information beyond a model based on point estimates since a relatively flat posterior distribution means that many models fit the data well and the most probable model is not particularly unique. On the other hand, a sharply peaked posterior distribution indicates that a point estimate solution might be a sound solution to use. In the discussion below we will start with *maximum likelihood* and *maximum a posteriori* approaches to unsupervised learning which use point estimates. Later we discuss approaches which give a full posterior distribution over models.

12.2.4 Maximum Likelihood and Maximum a Posteriori Solutions

With a *maximum likelihood* (ML) approach we derive a set of parameters that maximize the likelihood of the data given the model parameters, $p(D|\Theta)$. With the *maximum a posteriori* (MAP) approach we find the set of parameters that maximize the posterior, $p(\Theta|D)$, given the data. Thus, in terms of parameter-dependent probabilities, the MAP and ML solutions are related through Bayes rule $p(\Theta|D) \sim p(D|\Theta)p(\Theta)$. The MAP solution therefore enables us to include any prior knowledge we may have about the distribution of the parameter values.

The ease with which we may calculate $p(\Theta|D)$ depends on the functional forms of the likelihood and the prior. Also, if Θ is high dimensional, the evidence $p(D)$ may be difficult to evaluate. With a MAP solution only point estimates are used for the model parameters. These are denoted Θ_{MAP}, and they are based on the mode of the posterior distribution. Since the mode is unaffected when the distribution is multiplied by a constant, we can ignore the evidence in the denominator and only use the unnormalized distribution, which we denote $\widehat{p}(\Theta|D) = p(D|\Theta)p(\Theta)$. In general, though, we may need to evaluate the evidence, and this motivates our discussion of Monte Carlo methods below.

12.2.5 Variational Bayes

With a *variational Bayes* approach we can determine a posterior distribution over models. In this approach we approximate a posterior distribution $p(R|D)$ by a parameterized distribution $q(R)$. In this case D is the data, as before, and R is a parameter vector consisting of the model parameters Θ and any hidden variables within the model (for example, hidden labels assigning data components to clusters). Thus, we wish to make $q(R)$ as close as possible to the posterior $p(R|D)$. Since $p(R, D) = p(R|D)p(D)$ and $\int_{-\infty}^{\infty} q(R)\,\mathrm{d}R = 1$, we can write

$$\log\left[p(D)\right] = \log\left(\frac{p(R, D)}{p(R|D)}\right) \tag{12.49}$$

so

$$\begin{aligned}
\log\left[p(D)\right] &= \int_{-\infty}^{\infty} q(R)\log\left(\frac{q(R)p(R, D)}{q(R)p(R|D)}\right)\mathrm{d}R \\
&= \int_{-\infty}^{\infty} q(R)\log\left(\frac{p(R, D)}{q(R)}\right)\mathrm{d}R \\
&\quad + \int_{-\infty}^{\infty} q(R)\log\left(\frac{q(R)}{p(R|D)}\right)\mathrm{d}R \\
&= F\left[R\right] + KL\left[q||p\right] .
\end{aligned}$$

The second term

$$KL\left[q||p\right] = \int q(R) \log\left(\frac{q(R)}{p(R|D)}\right) \mathrm{d}R \qquad (12.50)$$

is a *Kullback–Leibler* (KL) divergence which quantifies the similarity of the two distributions $q(R)$ and $p(R|D)$. The key observation is that $\log\left[p(D)\right]$ does not depend on R. Since we want to minimize the KL divergence between $q(R)$ and $p(R|D)$, we can achieve this by optimizing $F[R]$, which is called the *variational free energy*. After making distributional and other assumptions we can derive an *expectation-maximization* (EM) algorithm which optimizes $F[R]$ and gives an effective approximation to the posterior distribution $p(R|D)$.

Though we do not give more detail here, the ML, MAP, and variational Bayes methods outlined above can be extended in many ways; For example, we could consider a *marginalized variational Bayes approach* in which we marginalize, or integrate out, further model parameters [12.51, 52]. With fewer parameters to estimate, this approach gives higher cluster accuracy but at the cost of model interpretability. We could consider *semisupervised* clustering [12.53] if *side information*, in the form of some sample labels, is present. Then again, whereas HCA may not be amenable to data integration, Bayesian methods could provide a route to unsupervised learning using multiple types of data; For example, with a *correspondence model* [12.54, 55], we can consider two types of data with one type of data believed dependent on the other. In a bioinformatics context an example would be gene expression array data which we believe may be partially influenced by microRNA expression [12.55].

12.2.6 Case Study 4: An Application to the Interpretation of Expression Array Data in Cancer Research

To illustrate an application of these methods we consider the use of ML, MAP, and variational Bayes methods to interpret gene expression array data derived in cancer research. We will start with a maximum likelihood or MAP solution giving point estimates for the model parameters. We start by specifying a model which includes making certain distributional assumptions for the data and model parameters. This leads us through to a likelihood bound which is optimized using an algorithmic procedure (an EM or expectation-maximization method).

As an example we consider latent process decomposition (LPD) [12.56]. We first make assumptions about the type of data we are considering. For expression array data, a reasonable assumption is that the gene expression measurements follow an approximate Gaussian distribution which will have a mean $\boldsymbol{\mu}$ and standard distribution $\boldsymbol{\sigma}$. Each sample in the data has a set of features labeled by an index $g = 1, \ldots, G$. Then, for feature g we draw a cluster index k ($k = 1, \ldots, K$) with a probability denoted β_k which selects a Gaussian with parameters μ_{gk} and σ_{gk}. Next we make assumptions about the probability distributions involved. For the $\boldsymbol{\beta}$ we assume a Dirichlet distribution, which is a standard assumption in this context. This Dirichlet distribution is parameterized by a K-dimensional vector $\boldsymbol{\alpha}$. The expression array data D consist of a set of samples, indexed $a = 1, \ldots, A$, each with a set of features labeled by g, and we thus denote the experimental measurements by e_{ga}. It is normal to work with the log of the likelihood so that we deal with summations rather than products. This is sound since the log-function is monotonic, so maximizing the log-likelihood is equivalent to maximizing the likelihood. The log-likelihood is then $\log p(D|\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\alpha})$, which can be factorized over the individual samples as

$$\log p(D|\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\alpha}) = \sum_{a=1}^{A} \log p(a|\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\alpha}) \,. \qquad (12.51)$$

We can rewrite this as a *marginalized* integral over the $\boldsymbol{\beta}$, that is,

$$\log p(D|\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\alpha}) = \sum_{a=1}^{A} \log \int p(a|\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\beta}) p(\boldsymbol{\beta}|\boldsymbol{\alpha}) \mathrm{d}\boldsymbol{\beta} \,, \qquad (12.52)$$

with $p(\boldsymbol{\beta}|\boldsymbol{\alpha})$ being the Dirichlet distribution. We introduce the Gaussian distributional assumption for the data

$$p(a|\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\beta}) = \prod_{g=1}^{G} \sum_{k=1}^{K} N\left(e_{ga}|k, \mu_{gk}, \sigma_{gk}\right) \beta_k \,. \qquad (12.53)$$

Exact inference with this model is intractable, so to increase the likelihood and therefore, implicitly, the probability of the model given the data, we follow the indirect route of deriving a lower bound on this log-likelihood via *Jensen's inequality*. We then maximize this lower bound using an iterative procedure based on

an expectation-maximization algorithm. Thus we get

$$\sum_{a=1}^{A} \log\left[p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\alpha})\right] = \sum_{a=1}^{A} \log \int_{\boldsymbol{\beta}} \left[\prod_{g=1}^{G}\sum_{k=1}^{K} N(e_{ga}|k,\mu_{gk},\sigma_{gk})\beta_k\right] p(\boldsymbol{\beta}|\boldsymbol{\alpha})\,\mathrm{d}\boldsymbol{\beta}\,. \tag{12.54}$$

If we define $\boldsymbol{E}_{p(z)} = \int z\,p(z)\mathrm{d}z$, then Jensen's inequality for a concave function $f(x)$ states that

$$f(\boldsymbol{E}_{p(z)}[z]) \geq \boldsymbol{E}_{p(z)}[f(z)]\,. \tag{12.55}$$

We have assumed a Dirichlet distribution for the β, so we could introduce a sample-specific (a-dependent) distribution for the $p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)$ as

$$E_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}[f(\boldsymbol{\beta})] = \int_{\boldsymbol{\beta}} f(\boldsymbol{\beta})\,p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)\,\mathrm{d}\boldsymbol{\beta}\,, \tag{12.56}$$

giving

$$\begin{aligned} & f\left[\int_{\boldsymbol{\beta}} p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)\boldsymbol{\beta}\,\mathrm{d}\boldsymbol{\beta}\right] \\ & = f(\boldsymbol{E}_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}[\beta]) \geq \boldsymbol{E}_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}[f(\boldsymbol{\beta})] \\ & = \int_{\boldsymbol{\beta}} f(\boldsymbol{\beta})\,p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)\,\mathrm{d}\boldsymbol{\beta} \end{aligned} \tag{12.57}$$

and so

$$\begin{aligned} & \sum_a \log\left[p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\alpha})\right] \\ & = \sum_a \log\left[\int_{\boldsymbol{\beta}} p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\beta})p(\boldsymbol{\beta}|\boldsymbol{\alpha})d\boldsymbol{\beta}\right] \\ & = \sum_a \log\left[\int_{\boldsymbol{\beta}} \left\{p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\beta})\frac{p(\boldsymbol{\beta}|\boldsymbol{\alpha})}{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\right\} p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)\,\mathrm{d}\boldsymbol{\beta}\right] \\ & = \sum_a \log\left[E_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\left\{p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\beta})\frac{p(\boldsymbol{\beta}|\boldsymbol{\alpha})}{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\right\}\right] \\ & \geq \sum_a E_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\left[\log\left\{p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\beta})\frac{p(\boldsymbol{\beta}|\boldsymbol{\alpha})}{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\right\}\right] \\ & = \sum_a E_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\left[\log\left\{p(a|\boldsymbol{\mu},\boldsymbol{\sigma},\boldsymbol{\beta})\right\}\right] \\ & \quad + \sum_a E_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\left[\log\left\{p(\boldsymbol{\beta}|\boldsymbol{\alpha})\right\}\right] \\ & \quad - \sum_a E_{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)}\left[\log\left\{p(\boldsymbol{\beta}|\boldsymbol{\gamma}_a)\right\}\right]\,. \end{aligned}$$

It is this lower bound which we optimize using a two-step expectation-maximization algorithm [12.56]. A variational Bayes approach leads to a similar iterative procedure to maximize the free energy term in (12.50).

We applied ML, MAP, and variational Bayes methods [12.55] to the interpretation of expression array data

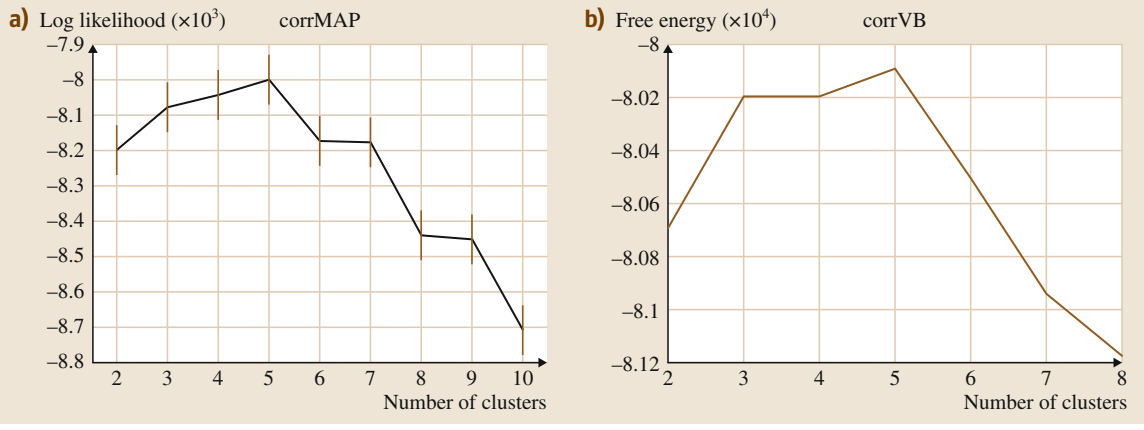

Fig. 12.7 **(a)** Estimated log-likelihood versus number of clusters using a MAP solution [12.55] for a breast cancer expression array dataset [12.57]. **(b)** Variational Bayes solution for the same dataset. For both methods the peak at five clusters indicates five principal subtypes of breast cancer. If more data were used, higher resolution may be achieved and we may observe more subtypes

derived from 78 primary breast cancer samples [12.57]. Using a MAP approach we obtained a solution for the model parameters using an EM algorithm. If we split the data into training and validation data, parameters in the log-likelihood can be estimated from the training set. Using this estimated log-likelihood and the validation data, we can then estimate model complexity, i. e., how many clusters are apparently present in the data – in this case how many subtypes of breast cancer are indicated. In Fig. 12.7a we show the estimated log-likelihood on left-out validation data for this breast cancer dataset. The peak indicates a minimum of five subtypes.

We also used a variational Bayes method with the same dataset [12.55]. In this case the maximum of the free energy versus number of clusters indicates the appropriate model complexity. In Fig. 12.7b the peak is also at 5, indicating that this is the most appropriate number of subtypes to consider. Interestingly, with variational Bayes, we do not need to use validation data.

Having found the most appropriate number of subtypes, we can use these methods to find those genes which are abnormally functioning within subtypes. The parameters μ_{gk} and σ_{gk} can be used to model the data distribution for gene g in cluster k. This is illustrated in Fig. 12.8 for two genes, *FOXA1* and *FOXC1*, which appear to be operating abnormally within one of these subtypes (cluster 5, denoted Cl5). The Gaussian distributions are determined by the (μ_{gk}, σ_{gk}), and the actual distribution of data values is shown below these Gaussian distributions. Whereas *FOXA1* and *FOXC1* appear to function normally in the other subtypes, *FOXA1* appears to be underexpressing and *FOXC1* overexpressing within this subtype.

As pointed out at the beginning, one further aspect in which these Bayesian unsupervised learning methods differ from HCA is that they allow for mixed membership. As a second example (Fig. 12.9), we apply a variational Bayes method to a lung cancer gene expression array dataset derived from 73 patient samples [12.59]. The peaks indicate the confidence that a particular patient belongs to a particular cluster. Many peaks are 1, but a number overlap several clusters, possibly indicating an unclear assignment of patient to subtype. We have used lung cancer as an illustration because there are a number of clinically established subtypes for this disease, such as small cell lung cancer and adenocarcinoma of the lung. The clinical assignments are indicated by the boundary markers in this plot, and there is reasonable agreement between clinical assignment to subtype and those assignments made by the variational Bayes method.

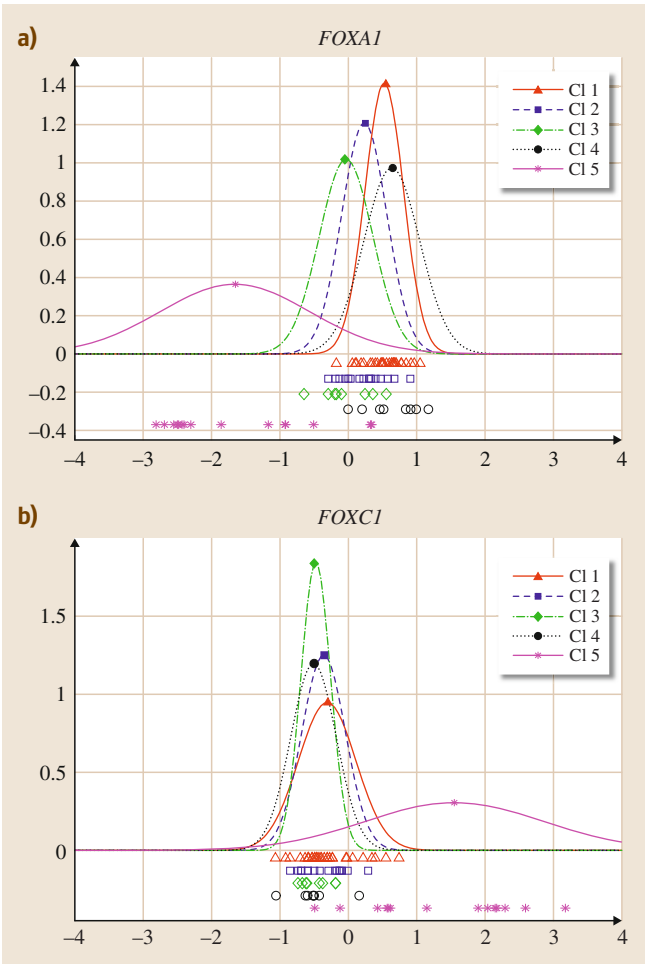

Fig. 12.8a,b Expression profiles for two genes within a subtype of breast cancer: *FOXA1* **(a)** and *FOXC1* **(b)**. This subtype can be identified with the basal-like or *basaloid* subtype of breast cancer. These two genes show a strong reciprocal anticorrelated expression profile within this subtype (after [12.58])

12.2.7 Monte Carlo Methods

Monte Carlo methods are potentially more exact than variational methods, and they are commonly used in probabilistic inference. If $\boldsymbol{\theta}$ is high dimensional, evaluating the evidence $\int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$ and finding the posterior $p(\boldsymbol{\theta}|D)$ is a difficult task. For the evidence, we could perform a *Monte Carlo integration*. Taken over an arbitrary distribution $h(\boldsymbol{\theta})$, we can perform Monte Carlo

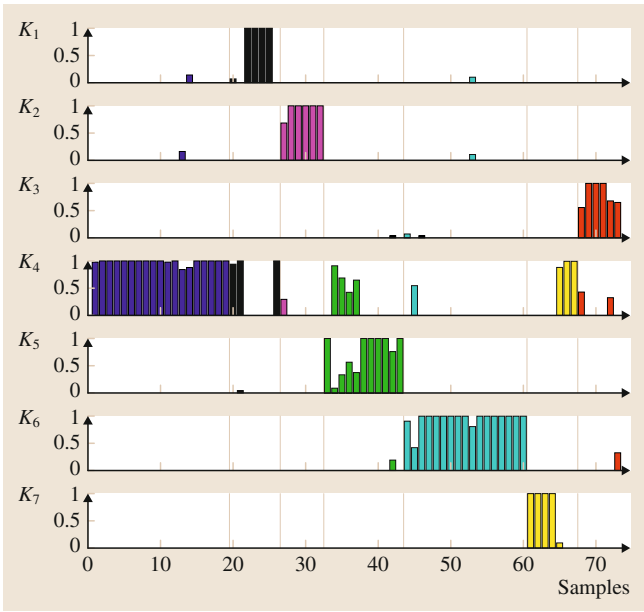

Fig. 12.9 The peaks indicate the confidence measure that a given sample is assigned to a particular cluster (see [12.58] for details about the derivation of this measure). The *fourth band down* has mainly adenocarcinoma of the lung (samples 1–20), and the plot indicates some confusion of assignment between this category and other subtypes of lung cancer

integration by writing $h(\boldsymbol{\theta}) = f(\boldsymbol{\theta})g(\boldsymbol{\theta})$ as

$$\int h(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = \int f(\boldsymbol{\theta})g(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} = E_{g(\boldsymbol{\theta})}\left[f(\boldsymbol{\theta})\right] \approx \frac{1}{M}\sum_{m=1}^{M} f\left(\boldsymbol{\theta}^{(m)}\right), \tag{12.58}$$

so that the integration becomes an expectation of $f(\boldsymbol{\theta})$ over $g(\boldsymbol{\theta})$. By deriving a number of observations $\boldsymbol{\theta}^{(m)}$ $(m = 1, \ldots, M)$, sampled from the distribution $g(\boldsymbol{\theta})$, and using these samples in $f(\boldsymbol{\theta})$, the original integral can be approximately evaluated. We could use this approach to evaluate the evidence and the posterior distribution by sampling from the unnormalized distribution $\widehat{p}(\boldsymbol{\theta}|D) = p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})$ to find the normalization $\int p(\boldsymbol{\theta}|D)\,\mathrm{d}\boldsymbol{\theta}$. To use this method of integration, we must be able to draw samples reliably from a distribution of choice, such as $\widehat{p}(\boldsymbol{\theta}|D)$. This is not easy, and here we briefly describe *Markov chain Monte Carlo* (MCMC) methods [12.60–62], based on the *Metropolis* algorithm, for performing this task.

A *Markov chain* is a sequence of samples $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots\}$ such that each sample is only dependent on the previous sample, i. e., $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}, \ldots, \boldsymbol{\theta}_1) = p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$, where t is the iteration index. In the MCMC approach, a *proposal* or *jump* distribution $Q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ is used to generate $\boldsymbol{\theta}_t$ from $\boldsymbol{\theta}_{t-1}$. With the *Metropolis algorithm* we assume that this distribution is symmetric, i. e., that $Q(\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t) = Q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t+1})$. Our objective is to reliably draw samples from a distribution $g(\boldsymbol{\theta})$, such as $p(\boldsymbol{\theta}|D)$. Thus, we start the procedure with some $\boldsymbol{\theta}_0$ such that $\widehat{g}(\boldsymbol{\theta}_0) > 0$. After a number of iterations this procedure will tend toward the stationary distribution $g(\boldsymbol{\theta})$ so that $\boldsymbol{\theta}_t$ represents a random draw from $g(\boldsymbol{\theta})$. In the standard approach to MCMC, to arrive at this stationary distribution, at each step a sample is selected from Q and either accepted or rejected based on a comparison with $g(\boldsymbol{\theta})$. Specifically, a candidate sample $\boldsymbol{\theta}^\star$ is sampled from $Q(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ and accepted with probability α_t given by

$$\alpha_t = \min\left(\frac{\widehat{g}(\boldsymbol{\theta}^\star)Q(\boldsymbol{\theta}_{t-1}|\boldsymbol{\theta}^\star)}{\widehat{g}(\boldsymbol{\theta}_{t-1})Q(\boldsymbol{\theta}^\star|\boldsymbol{\theta}_{t-1})}, 1\right) \tag{12.59}$$

or rejected. If $\boldsymbol{\theta}^\star$ is accepted, then $\boldsymbol{\theta}^\star$ is assigned to $\boldsymbol{\theta}_t$. Otherwise, if $\boldsymbol{\theta}^\star$ is rejected, then $\boldsymbol{\theta}_{t-1}$ is assigned to $\boldsymbol{\theta}_t$ instead. The Metropolis Hastings algorithm [12.63–65] extends the Metropolis algorithm to jump distributions which are not symmetric, and both of these methods are members of a broad class of approaches to sampling from high-dimensional distributions.

12.2.8 Case Study 5: Network Inference

The understanding of pathways and networks is crucial to our understanding of the functional organization of genes and proteins. At an abstract level a network can be viewed as a set of *nodes*, together with a set of directed or undirected *edges* between these nodes. Biological networks of interest include, for example, *transcriptional regulatory networks*. In this case a gene may express a protein that functions as a transcriptional inhibitor or, alternatively, as an activator of one or more target genes. In this case the genes can be viewed as the nodes of a network with the edges representing direct regulatory connections to other genes. With *signal transduction networks* the proteins are viewed as the nodes and the edges are corresponding protein–protein interactions. Further networks of interest are *metabolic networks*, where the metabolites are the nodes.

We could use unsupervised learning to determine the network structure, a task we could call *global inference of network topology*. However, suppose we consider a fully connected network with n nodes, then we are attempting to find $n(n-1)/2$ possible connec-

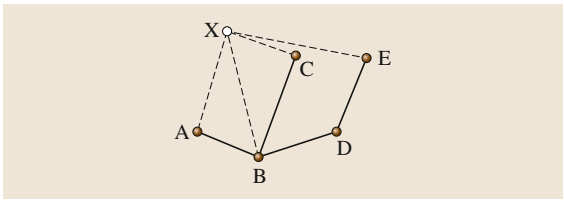

Fig. 12.10 Network completion. Using a training set of known links (*bold lines*) and nonlinks for nodes A–E we use supervised learning to predict links or nonlinks to a new node X (*dashed lines*)

tions given limited amounts of data. A cell may have tens of thousands of genes, whereas, in most experiments, we are only considering a few hundred samples with measurements corrupted by noise. Thus, the inference problem is typically highly underdetermined.

To make network inference more amenable as a machine learning task, a more tractable approach is *network completion* [12.66]. In this case we have an established pathway of interest and consequently we know certain links and nonlinks between pairs of nodes. The problem is therefore to determine whether a given node, perhaps representing a gene, has a link to this pathway or not. Since we have a training set of known links and nonlinks we can train a classifier via supervised learning and then predict a link to the pathway or otherwise, for the node of interest (Fig. 12.10). Various types of data are informative as to whether a functional link exists, and hence we can cast this supervised learning task as an application of the multiple kernel learning techniques of Sect. 12.1.4. As a supervised learning problem and with a smaller set of linkages to infer, this problem can give more reliable results than global unsupervised inference of network topology.

We can improve accuracy by reducing the search space further. Thus, as a third approach, we can consider the use of Bayesian methods to decide *the most probable network structure given a small set of candidate network topologies*. In this case it is assumed that a biologist has partially determined a network but remains undecided over a set of candidate topologies. Thus, the task is to determine which of these proposed network topologies is most probable, given the data. As an example, *Calderhead* et al. [12.67, 68] used Bayesian unsupervised learning to decide over alternative topologies proposed for the extracellular signal-regulated kinase (ERK) pathway. This signaling pathway regulates growth and is defective in many cancers. Binding of epidermal growth factor (EGF) to the epidermal growth factor receptor (EGFR) at the cell surface membrane activates ERK through a chain of proteins. Four candidate topologies were proposed. To compare individual pairs, representing different topologies, we use the *Bayes factor*, that is, the likelihood ratio

$$\frac{p(D|M_i)}{p(D|M_j)} \tag{12.60}$$

stated in terms of the *marginal likelihood* for model M to generate data D

$$p(D|M_i) = \int p(D|M_i, \boldsymbol{\theta}_i) p(\boldsymbol{\theta}_i) \mathrm{d}\boldsymbol{\theta}_i \,, \tag{12.61}$$

where the $\boldsymbol{\theta}_i$ are the set of model parameters marginalized or integrated out. Each protein in the chain is modeled using an ordinary differential equation (ODE), and we use optimization methods with a least-squares parameter fitting approach to find the optimal parameter values in these ODEs. We do not describe the ODEs here but refer to the original paper [12.67, 68]. Thus, a model can be written as $M = \{S, \boldsymbol{\theta}\}$, where S is the system of differential equations and $\boldsymbol{\theta}$ is the set of parameters. The marginal likelihood is therefore a nonlinear function based on the solution of these ODEs. It cannot be computed analytically, and so we must use MCMC-based methods. Because the posterior distribution is generated by complex dynamical systems models, it is necessary to use more sophisticated sampling methods than those mentioned in Sect. 12.2.7. Indeed, rather than static sampling distributions we use populations of annealed (temperature-dependent) distributions in an approach commonly referred to as population MCMC. Applied to ERK pathway models it was possible to use population MCMC to compute marginal likelihoods and thus Bayes factors, lending support to one of the proposed topology models for the ERK pathway [12.68].

12.3 Conclusions

Progress in the biomedical sciences can be furthered by improved data interpretation, in addition to the acquisition of more data. In this chapter we have seen that contemporary methods from machine learning can offer many advantages over more long-established data analysis methods. There are an increasing number of

studies where multiple types of data are acquired from the same sample. An example is the Cancer Genome Atlas [12.69] project, where multiple types of data are derived from the same tumor sample. In Sect. 12.1.3 we saw that many different types of data can be encoded into corresponding kernels and prediction can be achieved using multiple kernel learning. In Sect. 12.2 we saw that contemporary Bayesian unsupervised learning methods compare favorably with hierarchical cluster analysis, providing a confidence measure for assigning datapoints to clusters (Fig. 12.9) and an effective approach to model selection (Fig. 12.7). In the last section we saw that Bayesian methodology is the most effective approach to other tasks such as determining the most probable network topology given a set of candidate network topologies. Further innovation in machine learning will improve and expand the interpretability of many datasets from the biomedical sciences.

References

12.1 L. Bottou, O. Chapelle, D. DeCoste, J. Weston: *Large-Scale Kernel Machines*, Neural Information Processing Series (MIT Press, Cambridge 2007)

12.2 J. Platt, N. Cristianini, J. Shawe-Taylor: Large margin DAGS for multiclass classification, Adv. Neural Inform. Proces. Syst. **12**, 547–553 (2000)

12.3 Y. Lee, Y. Lin, G. Wahba: *Multicategory support vector machines, Technical Report 1043* (Univ. Madison, Wisconsin 2001)

12.4 T. Hastie, R. Tibshirani: Classification by pairwise coupling, Ann. Stat. **26**, 451–471 (1998)

12.5 T.G. Dietterich, G. Bakiri: Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. **2**, 263–286 (1995)

12.6 E.L. Allwein, R.E. Schapire, Y. Singer: Reducing multiclass to binary: A unifying approach for margin classifiers, J. Mach. Learn. Res. **1**, 133–141 (2000)

12.7 K.-B. Duan, S.S. Keerthi: Which is the best multiclass SVM Method? An empirical study, Proc. 6th Int. Workshop Multiple Classifier Syst. (2005), Vol. 3541 (Springer, Berlin, Heidelberg 2006) pp. 278–285

12.8 C. Cortes, V. Vapnik: Support vector networks, Mach. Learn. **20**, 273–297 (1995)

12.9 K. Veropoulos, C. Campbell, N. Cristianini: Controlling the sensitivity of support vector machines, Proc. Int. Joint Conf. Artif. Intell. (IJCAI) (1999)

12.10 J. Platt: Probabilistic outputs for support vector machines and comparison to regularised likelihood methods, Adv. Large Margin Classifiers (MIT Press, Cambridge 1999) pp. 61–74

12.11 A.E. Hoerl, R. Kennard: Ridge regression: Biased estimation for nonorthogonal problems, Technometrics **12**, 55–67 (1970)

12.12 C. Saunders, A. Gammermann, V. Vovk: Ridge regression learning algorithm in dual variables, Proc. Fifteenth Int. Conf. Mach. Learn. (ICML), ed. by J. Shavlik (Morgan Kaufmann, 1998)

12.13 V. Vapnik: *The Nature of Statistical Learning Theory* (Springer, New York 1995)

12.14 V. Vapnik: *Statistical Learning Theory* (Wiley, New York 1998)

12.15 B. Schölkopf, A.J. Smola: *Learning with Kernels* (MIT Press, Cambridge 2002)

12.16 J. Weston, A. Gammerman, M. Stitson, V. Vapnik, V. Vovk, C. Watkins: *Support vector density estimation*, Advances in Kernel Methods: Support Vector Machines (MIT Press, Cambridge 1998) pp. 293–306

12.17 A.J. Smola, B. Schölkopf: A tutorial on support vector regression, Stat. Comput. **14**, 199–222 (2004)

12.18 R.D. Williams, S.N. Hing, B.T. Greer, C.C. Whiteford, J.S. Wei, R. Natrajan, A. Kelsey, S. Rogers, C. Campbell, K. Pritchard-Jones, J. Khan: Prognostic classification of relapsing favourable histology Wilms tumour using cDNA microarray expression profiling and support vector machines, Genes Chromosom. Cancer **41**, 65–79 (2004)

12.19 I. Guyon, A. Elisseeff: An Introduction to Variable and Feature Selection, J. Mach. Learn. Res. **3**, 1157–1182 (2003)

12.20 T. Graepel, R. Herbrich, P. Bollmann-Sdorra, K. Obermayer: Classification on pairwise proximity data, Adv. Neural Inform. Proces. Syst. **11**, 438–444 (1998)

12.21 E. Pekalska, P. Paclik, R.P.W. Duin: A generalized kernel approach to dissimilarity based classification, J. Mach. Learn. Res. **2**, 175–211 (2002)

12.22 V. Roth, J. Laub, M. Kawanabe, J.M. Buhmann: Optimal cluster preserving embedding of nonmetric proximity data, IEEE Trans. Pattern Analys. Mach. Intell. **25**, 1540–1551 (2003)

12.23 R. Luss, A. d'Aspremont: Support vector machine classification with indefinite kernels, Adv. Neural Inform. Proces. Syst. **20**, 953–960 (2008)

12.24 Y. Ying, C. Campbell, M. Girolami: Analysis of SVM with Indefinite Kernels, Adv. Neural Informat. Proces. Syst. **22**, 2205–2213 (2009)

12.25 N. Cristianini, C. Campbell, J. Shawe-Taylor: Dynamically adapting kernels in support vector machines, Adv. Neural Inform. Proces. Syst. **11**, 204–210 (1999)

12.26 T. Joachims: Estimating the generalization performance of an SVM efficiently, Proc. 17th Int. Conf. Mach. Learn. (Morgan Kaufmann, 2000) pp. 431–438

12.27 O. Chapelle, V. Vapnik: Model selection for support vector machines, Adv. Neural Inform. Proces. Syst. **12**, 673–680 (2000)

12.28 V. Vapnik, O. Chapelle: Bounds on error expectation for support vector machines, Neural Comput. **12**, 2013–2036 (2000)

12.29 P. Sollich: Bayesian methods for support vector machines: Evidence and predictive class probabilities, Mach. Learn. **46**, 21–52 (2002)

12.30 J. Shawe-Taylor, N. Cristianini: *Kernel Methods for Pattern Analysis* (Cambridge Univ. Press, Cambridge 2004)

12.31 H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins: Text classification using string kernels, J. Mach. Learn. Res. **2**, 419–444 (2002)

12.32 C. Leslie, R. Kuang: Fast kernels for inexact string matching, 16th Ann. Conf. Learning Theory 7th Kernel Workshop, Vol. 2777 (Springer, Berlin, Heidelberg 2003) pp. 114–128

12.33 S. Vishwanathan, A. Smola: Fast Kernels for String and Tree Matching, Adv. Neural Inform. Proces. Syst. **15**, 569–576 (2003)

12.34 I.R. Kondor, J.D. Lafferty: Diffusion kernels on graphs and other discrete structures, Proc. Int. Conf. Mach. Learn. (Morgan Kaufmann, San Francisco, 2002) pp. 315–322

12.35 A.J. Smola, I.R. Kondor: Kernels and regularization on graphs, Conf. Learning Theory (COLT), Vol. 2777 (Springer, Berlin, Heidelberg 2003) pp. 144–158

12.36 T. Gartner, P. Flach, S. Wrobel: On graph kernels: Hardness results and efficient alternatives, Proc. Annu. Conf. Computational Learning Theory (COLT) (Springer, Berlin, Heidelberg 2003) pp. 129–143

12.37 S.V.N. Vishwanathan, K.M. Borgwardt, I.R. Kondor, N.N. Schraudolph: Graph Kernels, J. Mach. Learn. Res. **9**, 1–41 (2008)

12.38 G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M.I. Jordan: Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. **5**, 27–72 (2004)

12.39 F. Bach, G.R.G. Lanckriet, M.I. Jordan: Multiple kernel learning, conic duality and the SMO algorithm, Proc. 21st Int. Conf. Machine Learning (ICML) (Morgan Kaufmann, New York 1998)

12.40 S. Sonnenburg, G. Rätsch, C. Schäfer, B. Schölkopf: Large scale multiple kernel learning, J. Mach. Learn. Res. **7**, 1531–1565 (2006)

12.41 A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet: SimpleMKL, J. Mach. Learn. Res. **9**, 2491–2521 (2008)

12.42 Z. Xu, R. Jin, I. King, M.R. Lyu: An extended level method for multiple kernel learning, Adv. Neural Inform. Proces. Syst. **22**, 1825–1832 (2008)

12.43 Y. Ying, K. Huang, C. Campbell: Enhanced protein fold recognition through a novel data integration approach, BMC Bioinf. **10**, 267–285 (2009)

12.44 T. Damoulas, M. Girolami: Probabilistic multi-class multi-kernel learning: On protein fold recognition and remote homology detection, Bioinformatics **24**, 1264–1270 (2008)

12.45 G.R.G. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, W.S. Noble: A statistical framework for genomic data fusion, Bioinformatics **20**, 2626–2635 (2004)

12.46 M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, A. Zien: Efficient and accurate lp-norm multiple kernel learning, Adv. Neural Inform. Proces. Syst. **22**, 997–1005 (2009)

12.47 U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, A.J. Levine: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Proc. Natl. Acad. Sci. USA **96**(12), 6745–6750 (1999)

12.48 B. Everitt: *Cluster Analysis* (Arnold, New York 1993)

12.49 L. Kaufman, P.J. Rousseeuw: *Finding Groups in Data* (Wiley, New York 2005)

12.50 R.O. Duda, P.E. Hart, D.G. Stork: *Pattern classification* (Wiley, New York 2001)

12.51 Y.W. Teh, D. Newman, M. Welling: A collapsed variational Bayesian inference algorithm for latent dirichlet allocation, Adv. Neural Inform. Proces. Syst. **19**, 1353–1360 (2006)

12.52 Y. Ying, P. Li, C. Campbell: A marginalized variational Bayesian approach to the analysis of array data, BMC Proc. **2**(4), S7 (2008)

12.53 P. Li, Y. Ying, C. Campbell: A variational approach to semi-supervised clustering, Proc. ESANN2009 (2009) pp. 11–16

12.54 D.M. Blei, M.I. Jordan: Modeling annotated data, Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. (ACM Press, New York 2003) pp. 127–134

12.55 P. Agius, Y. Ying, C. Campbell: Bayesian Unsupervised Learning with Multiple Data Types, Stat. Appl. Genet. Molec. Biol. **8**, 27 (2009)

12.56 S. Rogers, M. Girolami, C. Campbell, R. Breitling: The latent process decomposition of cdna microarray datasets, IEEE/ACM Trans. Comput. Biol. Bioinforma. **2**, 143–156 (2005)

12.57 C. Blenkiron, L.D. Goldstein, N.P. Thorne, I. Spiteri, S.F. Chin, M.J. Dunning, N.L. Barbosa-Morais, A.E. Teschendorff, A.R. Green, I.O. Ellis, S. Tavaré, C. Caldas, E.A. Miska: MicroRNA expression profiling of human breast cancer identifies new markers of tumour subtype, Genome Biol. **8**(10), R214–1–R214–16 (2007)

12.58 L. Carrivick, S. Rogers, J. Clark, C. Campbell, M. Girolami, C. Cooper: Identification of prognostic signatures in breast cancer microarray data using Bayesian techniques, J. R. Soc. Interf. **3**, 367–381 (2006)

12.59 E. Garber, O.G. Troyanskaya, K. Schluens, S. Petersen, Z. Thaesler, M. Pacyna-Gengelbach, M. van de Rijn, G.D. Rosen, C.M. Perou, R.I. Whyte, R.B. Altman, P.O. Brown, D. Botstein, I. Petersen: Diversity of gene expression in adenocarcinoma of the lung, Proc. Natl. Acad. Sci. USA **98**, 13784–13789 (2001)

12.60 C. Andrieu, N. De Freitas, A. Doucet, M.I. Jordan: An introduction to MCMC for machine learning, Mach. Learn. **50**, 5–43 (2003)
12.61 W.R. Gilks, S. Richardson, D.J. Spiegelhalter: *Markov Chain Monte Carlo in Practice* (Chapman Hall/CRC, New York 1996)
12.62 C.P. Robert, G. Casella: *Monte Carlo Statistical Methods* (Springer, Berlin, Heidelberg 2004)
12.63 S. Chib, E. Greenberg: Understanding the Metropolis Hastings Algorithm, Am. Stat. **49**(4), 327–335 (1995)
12.64 B.A. Berg: *Markov Chain Monte Carlo Simulations and Their Statistical Analysis* (World Scientific, Singapore 2004)
12.65 W.M. Bolstad: *Understanding Computational Bayesian Statistics* (Wiley, New York 2010)
12.66 K. Bleakley, G. Biau, J.-P. Vert: Supervised reconstruction of biological networks with local models, Bioinformatics **23**, i57–i65 (2007)
12.67 B. Calderhead, M. Girolami: Estimating Bayes factors via thermodynamic integration and population MCMC, Comput. Stat. Data Anal. **53**, 4028–4045 (2009)
12.68 T.R. Xu, V. Vyshemirsky, A. Gormand, A. von Kriegsheim, M. Girolami, G.S. Baillie, D. Ketley, A.J. Dunlop, G. Milligan, M.D. Houslay, W. Kolch: Inferring signaling pathway topologies from multiple perturbation measurements of specific biochemical species, Sci. Signal. **3**(113), ra20:1–ra20:10 (2010)
12.69 Cancer Genome Atlas: Available at http://cancergenome.nih.gov